Nicht für den Prüfling bestimmt

I Erläuterungen

Voraussetzungen gemäß KCBG und Abiturerlassen BG jeweils in der für den Abiturjahrgang geltenden Fassung

Standardbezug
Die nachfolgend ausgewiesenen Kompetenzbereiche sind für die Bearbeitung der jeweiligen Aufgabe besonders bedeutsam. Darüber hinaus können weitere, hier nicht ausgewiesene Kompetenzbereiche für die Bearbeitung der Aufgabe nachrangig bedeutsam sein, zumal die Kompetenzbereiche in engem Bezug zueinanderstehen. Die Operationalisierung des Bezugs zu den Kompetenzbereichen des Standardbezugs erfolgt in Abschnitt II.

Aufgabe	Kompetenzbereiche				
	K1	K2	K3	K4	K5
1.1.1	X			X	
1.1.2	X				
1.2	X	X			
1.3.1		X		X	
1.3.2			X		
1.3.3				X	
1.4.1		X	X		
1.4.2		X		X	
1.4.3				X	
2.1	X	X			
2.2.1		X			
2.2.2		X			
2.2.3				X	
2.2.4				X	
2.2.5				X	
2.3			X		
2.4	X				X

Inhaltlicher Bezug
Die nachfolgend ausgewiesenen Themenfelder sind die wesentliche inhaltliche Grundlage für die vorliegenden Aufgaben. Darüber hinaus können weitere, hier nicht explizit ausgewiesene Themenfelder für die Bearbeitung nachrangig bedeutsam sein.
Q1: Objektorientierte Softwareentwicklung
Q2: Datenbanksysteme
Q3: Datenkommunikation
verbindliche Themenfelder: Objektorientierte Modellierung (Q1.1), Implementierung von Klassen und Assoziationen (Q1.2), Konzeptionelle und logische Modellierung einer Datenbank (Q2.1), Datenabfrage und Datenmanipulation mit SQL (Q2.2), Serielle Kommunikation (Q3.1)

II Lösungshinweise

In den nachfolgenden Lösungshinweisen sind alle wesentlichen Gesichtspunkte, die bei der Bearbeitung der einzelnen Aufgaben zu berücksichtigen sind, konkret genannt und diejenigen Lösungswege aufgezeigt, welche die Prüflinge erfahrungsgemäß einschlagen werden. Selbstverständlich sind jedoch Lösungswege, die von den vorgegebenen abweichen, aber als gleichwertig betrachtet werden können, ebenso zu akzeptieren.

Aufg.	**erwartete Leistungen**	**BE**		
		I	**II**	**III**
1.1.1	beschreiben, nennen − Analyse und Definition der Anforderungen Die Dienstleistungen, Einschränkungen und Ziele des Systems werden in Zusammenarbeit mit den Systembenutzern aufgestellt. Dann werden sie detaillierter definiert und dienen so als Systemspezifikation. (Produkte z.B.: Lastenheft, Pflichtenheft, Meilensteine, UML-Anwendungsfalldiagramm, Prototyp Benutzeroberfläche, Testdaten) − Softwareentwurf Beim Softwareentwurf geht es um das Erkennen und Beschreiben der grundlegenden abstrakten Softwaresysteme und ihrer Beziehung zueinander. (Produkte z.B.: UML-Klassendiagramm, UML-Sequenzdiagramm, Modell der Datenhaltung, Schnittstellenspezifikation, Entwicklerdokumentation) − Implementierung und Komponententests In dieser Phase wird der Softwareentwurf in einer Reihe von Programmen oder Programmeinheiten umgesetzt. Das Testen der Einheiten stellt sicher, dass jede Einheit ihre Spezifikationen erfüllt. (Produkte z.B.: kommentierte Programmmodule, Testprotokolle) − Integration und Systemtests Die einzelnen Programme oder Programmeinheiten werden zusammengeführt bzw. integriert und als Ganzes getestet, um sicherzustellen, dass die Softwareanforderungen erfüllt werden. Nach den Tests wird das Softwaresystem an den Kunden ausgeliefert. (Produkte z.B.: Testprotokolle, Dokumentation, Benutzerhandbuch) − Betrieb und Wartung Das System wird installiert und zum Gebrauch frei gegeben. Zur Wartung gehören das Korrigieren von Fehlern, die in den früheren Phasen nicht entdeckt wurden, die Verbesserung der Implementierung von Systemeinheiten und die Verbesserung des Systems, falls neue Anforderungen entdeckt werden. (Produkte z.B.: Benutzerhandbuch, Abnahmeprotokoll, Dokumentation)			
	beschreiben	2	3	
	nennen	2		
1.1.2	beschreiben − Funktionalität: Die Funktionen erfüllen die Anforderungen hinsichtlich Richtigkeit, Sicherheit und Angemessenheit. − Zuverlässigkeit: Fähigkeit der Software, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren. Dazu gehören Verfügbarkeit, Fehlertoleranz und Wiederherstellbarkeit. − Benutzbarkeit: Die Gebrauchstauglichkeit der Software hinsichtlich Attraktivität, Bedienbarkeit und Verständlichkeit unterliegt der individuellen Beurteilung durch die Personen, die mit der Software arbeiten. − Wartbarkeit: Aufwand, der zur Durchführung vorgegebener Änderungen notwendig ist. Änderungen können Korrekturen, Verbesserungen oder Anpassungen an Änderungen der Umgebung, der Anforderungen oder der funktionalen Spezifikationen einschließen. Hinweis: Die Beschreibung anderer Qualitätskriterien ist als gleichwertig zu betrachten.	4		

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE		
		I	II	III
1.2	beschreiben − Ein System wird durch die Summe aller Anwendungsfälle beschrieben. Diese werden durch die Systemgrenze zusammengefasst. Der Name des Systems steht innerhalb der Systemgrenze. − Ein Akteur modelliert einen Typ oder eine Rolle, die ein externer Benutzer oder ein externes System während der Interaktion mit dem System einnimmt. − Ein Anwendungsfall wird durch eine Ellipse innerhalb der Systemgrenze dargestellt. In der Ellipse steht die Funktionsbeschreibung. − Die Akteure werden über Verbindungslinien (Beziehungen) mit den Anwendungsfällen verbunden. Eine Beziehung zwischen einem Akteur und einem Anwendungsfall besagt, dass der Akteur am Anwendungsfall beteiligt ist oder diesen Anwendungsfall auslösen kann. − Eine include-Beziehung modelliert die unbedingte Einbindung der Funktionalität eines Anwendungsfalls in einen anderen Anwendungsfall (muss-Beziehung). Dargestellt wird die include-Beziehung durch einen gestrichelten Pfeil, wobei der Pfeil auf den inkludierten Anwendungsfall zeigt. − Eine extend-Beziehung modelliert die bedingte Einbindung der Funktionalität eines Anwendungsfalls in einen anderen Anwendungsfall (kann-Beziehung). Der erweiterte Anwendungsfall wird durch sogenannte Erweiterungspunkte bzw. Extension Points detailliert beschrieben. Die Bedingung wird definiert. Tritt die Bedingung ein, wird die Erweiterung ausgeführt. Dargestellt wird die extend-Beziehung durch einen gestrichelten Pfeil, wobei der Pfeil auf den erweiterten Anwendungsfall zeigt.	3		
	erläutern Im System „Taxizentrale“ löst der Kunde den Anwendungsfall „Taxi bestellen“ durch Anruf bei der Taxizentrale aus. Der Mitarbeiter ist an diesem Anwendungsfall beteiligt, da er den Anruf entgegennimmt. Der Anwendungsfall „Taxi bestellen“ bindet die Funktionalität des Anwendungsfalls „Auftrag mit Ortsangaben erfassen“ zwingend ein. Der Mitarbeiter kann die Verfügbarkeit der Taxis prüfen. Wenn ein Taxi verfügbar ist, kann der Mitarbeiter die Bereitschaft zur Übernahme des Auftrags bei dem Fahrer des Taxis anfragen. Signalisiert der Fahrer seine Bereitschaft, kann der Mitarbeiter den Auftrag an den Fahrer vergeben.		2	

Aufg.	erwartete Leistungen	BE I	BE II	BE III

1.3.1	überführen, implementieren
	`public class Auftrag {`
	`  private int auftragNr;`
	`  private Adresse von;`
	`  private Adresse nach;`
	`  private DateTime start;`
	`  private DateTime ende;`
	`  private double fahrstrecke;`
	`  private double fahrpreis;`
	`  private Fahrer fahrer;`
	`  private Taxi taxi;`
	`  private static int autowert = 0;`
	
	`  public Auftrag(Taxi t, Fahrer f, String auftragsdaten) {`
	`    auftragNr = ++autowert;`
	`    taxi = t;`
	`    fahrer = f;`
	`    String[] werte = auftragsdaten.split(";");`
	`    String[] adresse = werte[0].split(",");`
	`    von = new Adresse(adresse[0],adresse[1],adresse[2]);`
	`    adresse= werte[1].split(",");`
	`    nach = new Adresse(adresse[0],adresse[1],adresse[2]);`
	`    start = new DateTime(werte[2]);`
	`    ende = new DateTime(werte[3]);`
	`    fahrstrecke = Double.parseDouble(werte[4]);`
	`    fahrpreis = Double.parseDouble(werte[5]);`
	`  }`
	`}`
	überführen
	implementieren
		
		2
			
			3	

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.3.2	entwickeln, zeichnen			
	: TaxiZentrale taxi : Taxi bearbeiteAuftrag(von, nach) getTaxisSortiertNachStandort(von) {taxiListe} loop [für jedes taxi aus taxis und Auftrag nicht vergeben] getFahrer() { f } anfragenBereitschaft(f, von, nach) { ok } opt [ok = true] vergebeAuftrag(f, von, nach) { ok }			
	entwickeln		3	2
	zeichnen	3		
1.3.3	implementieren (siehe Code)		3	7

```
public List<Taxi> getTaxisSortiertNachStandort(Adresse von){
  List<Taxi> sortierteListe = new List<Taxi>();
  List<Taxi> freieTaxis = new List<Taxi>();
  for(Taxi t : alleTaxis) {
    if(t.getStatus() == Taxi.FREI) {
      int entfernung1 = t.ermittleAktuellenStandort().
                              ermittleEntfernungNach(von);
      int i = 0;
      for( ; i < freieTaxis.size(); i++) {
        int entfernung2 = freieTaxis.get(i).
                            ermittleAktuellenStandort().
                              ermittleEntfernungNach(von);
        if(entfernung1 < entfernung2) {
          break;
        }
      }
```

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE		
		I	II	III

```
      freieTaxis.add(i, t);
    }
  }
  int zaehler = 0;
  for(int i = 0; i < freieTaxis.size() && zaehler < 5; i++){
    sortierteListe.add(freieTaxis.get(i));
    zaehler++;
  }
  return sortierteListe;
}
```

Aufg.	erwartete Leistungen	I	II	III
1.4.1	entwickeln, zeichnen			

calculatePrice(distance: double)

price := GRUNDPREIS

distance <= 15 ?

J: price := price + distance*TARIF1

N:
price := price + 15*TARIF1
distance := distance - 15
price := price + distance*TARIF2

Rückgabe: price

Aufg.	erwartete Leistungen	I	II	III
	entwickeln		3	
	zeichnen	2		

<table>
<tr><th rowspan="2">Aufg.</th><th rowspan="2">erwartete Leistungen</th><th colspan="3">BE</th></tr>
<tr><th>I</th><th>II</th><th>III</th></tr>
<tr><td>1.4.2</td><td>implementieren
<pre>public int writeData(String data) {
  //Schnittstelle nicht geöffnet oder Datenträger fehlt
  int fehlerStatus = 2;
  if(serial.open() && data.length() > 0) {
    data = data + DLE + ETX +
           calculateCheckDigit(data);
    char answer;
    do {
      writeChar(STX);
      if(readChar() == DLE) {
        serial.write(data.getBytes(), data.length());
        answer = readChar();
        if(answer == DLE) {
          fehlerStatus = 0; //ohne Fehler
        } else if(answer == EM) {
          fehlerStatus = 1; //Datenträger voll
          break;
        }
      } else {
        break;
      }
    } while(answer == NAK);
  }
  serial.close();
  return fehlerStatus;
}</pre></td><td></td><td>2</td><td>4</td></tr>
</table>

<table>
<tr><th rowspan="2">Aufg.</th><th rowspan="2">erwartete Leistungen</th><th colspan="3">BE</th></tr>
<tr><th>I</th><th>II</th><th>III</th></tr>
<tr><td>1.4.3</td><td>implementieren
<pre><code>private void run() {
  while(!eks.isKeyAvailable());
  driverNumber = eks.readDriverNumber();
  double km = 0, price = 0;
  String from = "", to = "", start="", end="";
  String orderData = "";
  while(eks.isKeyAvailable()) {
    int button = readButtons();
    switch(button) {
      //Taxameter auf FREI schalten
      case 1: taxi.setStatus(Taxi.FREI);
        orderData = "";
        this.setTaxiSign(true);
        break;
      //Auftrag angenommen, Anfahrt Kunde
      case 2: taxi.setStatus(Taxi.ANFAHRT_KUNDE);
        this.setTaxiSign(false);
        break;
      //Beginn Kundenfahrt
      case 3: km = taxi.getKmStand();
        from = taxi.ermittleAktuellenStandort().toString();
        start = new DateTime().toString();
        taxi.setStatus(Taxi.ANFAHRT_ZIEL);
        this.setTaxiSign(false);
        break;
      //Kundenfahrt beenden
      case 4: km = taxi.getKmStand() - km;
        price = this.calculatePrice(km);
        to = taxi.ermittleAktuellenStandort().toString();
        end = new DateTime().toString();
        orderData = taxi.getWagenNr() + ";" + driverNumber +
          ";" + from + ";" + to + ";" + start + ";" + end +
          ";" + km + ";" + price;
        eks.writeData(orderData);
        display.printLine("" + price);
        break;
      //Quittung drucken
      case 5: if(!orderData.equals(""))
        thermalPrinter.printReceipt(orderData);
    }
  }
  driverNumber = -1;
  taxi.setStatus(Taxi.NICHT_IN_BETRIEB);
}</code></pre></td><td></td><td>5</td><td>5</td></tr>
<tr><td></td><td>Summe 60</td><td>18</td><td>24</td><td>18</td></tr>
</table>

Aufg.	erwartete Leistungen	BE I	BE II	BE III
2.1	überführen Fahrer(fahrerNr, nachname, vorname, stundenlohn, iban) Taxi(wagenNr, kennzeichen) Einsatz(eID, beginn, ende, fahrerNr#, wagenNr#) Auftrag(auftragNr, start, ende, eID#, von#, nach#) Adresse(aID, strasse, plz, ort, oID#) Ortsteil(oID, bezeichnung) begründen Alle Entitätstypen werden mit ihren Attributen in Relationen überführt. Der Name des Fahrers muss zur Einhaltung der ersten Normalform aufgespalten werden. Bei 1:n-Beziehungen wie zum Beispiel beinhaltet zwischen Einsatz und Auftrag wird der PK der 1-Relation (eID) als FK in die n-Relation aufgenommen. Die Nichtschlüsselattribute sind vollständig vom jeweiligen PK abhängig und es existieren keine transitiven Abhängigkeiten, so dass die 3. Normalform eingehalten wird.	4	2	
2.2.1	formulieren INSERT INTO Fahrer VALUES(333, 'Knebel', 'Lennard', 12.0, null); INSERT INTO Einsatz(beginn, ende, fahrerNr, wagenNr) VALUES('2023-05-18 08:00:00', '2023-05-18 17:00:00', 333, (SELECT wagenNr FROM Taxi WHERE kennzeichen = 'F-TT 661'));		5	
2.2.2	formulieren UPDATE Einsatz SET fahrerNr = 333 WHERE fahrerNr = (SELECT fahrerNr FROM Fahrer WHERE nachname = 'Tischendorf' AND vorname = 'Tobias') AND beginn BETWEEN '2023-05-01%' AND '2023-05-14%';		3	
2.2.3	entwickeln SELECT * FROM Taxi WHERE wagenNr NOT IN (SELECT wagenNr FROM Einsatz WHERE beginn LIKE '2023-05-30%' OR ende LIKE '2023-05-30%');		1	2
2.2.4	entwickeln SELECT CONCAT(e.beginn," - ", e.ende) AS 'Arbeitszeit von - bis', TIMESTAMPDIFF(MINUTE, e.beginn, e.ende)/60 AS 'Anzahl Stunden' FROM Einsatz e JOIN Fahrer f USING(fahrerNr) WHERE e.beginn LIKE '2023-03%' AND f.nachname = 'Großfuss' AND f.vorname = 'Karl-Heinz';		2	2
2.2.5	implementieren SELECT f.nachname, f.vorname, f.iban, SUM(TIMESTAMPDIFF(MINUTE, e.beginn, e.ende)/60 * f.stundenlohn) AS 'Löhne März 2023' FROM Einsatz e JOIN Fahrer f USING(fahrerNr) WHERE e.ende LIKE '2023-03%' GROUP BY f.fahrerNr ORDER BY f.nachname, f.vorname;		1	3

Aufg.	**erwartete Leistungen**	**BE**		
		I	**II**	**III**
2.3	entwickeln, zeichnen erstzulassung kilometerstand art stelle wagenNr Taxi [1,1] hat [0,n] Schaden datum [0,n] sID [1,1] gehört zu erfordert rID mID [1,1] plaetze [0,1] Modell kosten Reparatur datum bezeichnung hersteller [0,n] [0,n] verwendet führt durch [1,2] [1,1] name aID Antriebsart wID Werkstatt adresse [1,1] bezeichnung hat apID [1,n] Ansprechpartner name telefon entwickeln zeichnen	4	3	3
2.4	nennen Die Tabelle weist Redundanzen auf, beispielsweise die Telefonnummer der Fahrer. Diese wird unnötig mehrfach gespeichert. Hinzu kommt, dass es zu Inkonsistenzen kommen kann, wenn eine Telefonnummer fehlerhaft geschrieben wird, was bereits bei Herrn Müller in der vorliegenden Tabelle der Fall ist.	2		
	beschreiben Mögliche Anomalien sind: Änderungsanomalie: Bekommt ein Fahrer eine neue Telefonnummer, muss sie an mehreren Stellen geändert werden. Wird das nicht konsequent gemacht, kommt es zu einem inkonsistenten Datenbestand. Einfügeanomalie: Ein neues Fahrzeug kann nur eingefügt werden, wenn es schon einem Fahrer oder einer Fahrt zugeordnet ist. Löschanomalie: Wird der Fahrer Finn Kager gelöscht, gehen die Daten des Fahrzeugs Mercedes Vito unbeabsichtigt ebenfalls verloren.	2	1	
	Summe 40	**12**	**18**	**10**

III Bewertung und Beurteilung

Die Bewertung und Beurteilung erfolgt unter Beachtung der nachfolgenden Vorgaben nach § 33 der Oberstufen- und Abiturverordnung (OAVO) in der jeweils geltenden Fassung. Bei der Bewertung und Beurteilung der sprachlichen Richtigkeit in der deutschen Sprache sind die Bestimmungen des § 9 Abs. 12 Satz 3 OAVO in Verbindung mit Anlage 9b anzuwenden.

Bei der Bewertung und Beurteilung der Übersetzungsleistung in den Fächern Latein und Altgriechisch sind die Bestimmungen des § 9 Abs. 14 OAVO in Verbindung mit Anlage 9c anzuwenden.

Der Fehlerindex ist nach Anlage 9b zu § 9 Abs. 12 OAVO zu berechnen. Für die Ermittlung der Punkte nach Anlage 9a zu § 9 Abs. 12 OAVO sowie Anlage 9c zu § 9 Abs. 14 OAVO wird jeweils der ganzzahlige nicht gerundete Prozentsatz bzw. Fehlerindex zugrunde gelegt.

Für die Bewertung in den modernen Fremdsprachen ist der „Erlass zur Bewertung und Beurteilung von schriftlichen Arbeiten in allen Grund- und Leistungskursen der neu beginnenden und fortgeführten modernen Fremdsprachen in der gymnasialen Oberstufe, dem beruflichen Gymnasium, dem Abendgymnasium und dem Hessenkolleg“ vom 7. August 2020 (ABl. S. 519) zugrunde zu legen. Demnach erfolgt die Bewertung und Beurteilung mit der Maßgabe, dass lediglich bei der Ermittlung des Prüfungsergebnisses (Note) aus Prüfungsteil 1 und 2 gerundet wird.

Darüber hinaus sind die Vorgaben der Erlasse „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen (Abiturerlass)“, „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen im beruflichen Gymnasium (fachrichtungs-/ schwerpunktbezogene Fächer) (Abiturerlass BG)“ und „Durchführungsbestimmungen zum Landesabitur“ in der für den Abiturjahrgang geltenden Fassung zu beachten.

Als Kriterien für die Bewertung und Beurteilung dienen unter Beachtung der Zielsetzung der gymnasialen Oberstufe nach § 1 Abs. 2 OAVO neben dem Inhaltlichen auch die in den Kerncurricula genannten überfachlichen Kompetenzen, insbesondere die Sprachkompetenz und Wissenschaftspropädeutik; dies zeigt sich u.a. in qualitativen Merkmalen wie Strukturierung, Differenziertheit, (fach-)sprachlicher Gestaltung und Schlüssigkeit der Argumentation.

Im Fach Praktische Informatik besteht die Prüfungsleistung aus der Bearbeitung eines Vorschlags, wofür insgesamt maximal 100 BE vergeben werden können. Ein Prüfungsergebnis von **5 Punkten (ausreichend)** setzt voraus, dass mindestens 45% der zu vergebenden BE erreicht werden. Ein Prüfungsergebnis von **11 Punkten (gut)** setzt voraus, dass mindestens 75% der zu vergebenden BE erreicht werden.

Gewichtung der Aufgaben und Zuordnung der Bewertungseinheiten zu den Anforderungsbereichen für die Interpretationsaufgabe

Aufgabe	**Bewertungseinheiten in den Anforderungsbereichen**			**Summe**
	AFB I	**AFB II**	**AFB III**	
1	18	24	18	**60**
2	12	18	10	**40**
Summe	**30**	**42**	**28**	**100**

Die auf die Anforderungsbereiche verteilten Bewertungseinheiten innerhalb der Aufgaben sind als Richtwerte zu verstehen.